

Learn **OpenXava** by example

JPA

JUnit

Liferay

Hibernate Validator

PostgreSQL

Eclipse

HtmlUnit

Javier Paniza

OPENXAVA

Learn
OpenXava
by example

EDITION 1.1.1

© 2011 *Javier Paniza*

About the book

This book is meant to teach you how to develop Java Enterprise applications with OpenXava as well as other Java related technologies, tools and frameworks. Together we will develop step by step a complete application from scratch.

Why a book about OpenXava? There is already plenty of free documentation available, which is very exhaustive and always up to date. However, some members of the OpenXava community kept asking me for a book, so eventually I was convinced that writing one would be a good idea.

The OpenXava documentation describes all syntax and associated semantics of OpenXava precisely. Without a doubt, it's an essential tool, and allows you to learn all the OpenXava secrets. But, if you are a newcomer to the Java Enterprise world, you might need more than reference documentation. You want a lot of code examples as well as a means to learn the other related technologies which are needed to create an advanced application.

In this book you'll learn not only about OpenXava, but JPA, Eclipse, PostgreSQL, JUnit, HtmlUnit, Hibernate Validator framework, Liferay, etc. as well. And even more important: you're going to learn techniques to fulfill the common and advanced requirements you will be facing in typical business applications.

Don't hesitate to contact me if you have any suggestions about the book at javierpaniza@yahoo.com. If you find grammatical or orthographic errors I will refund your money¹.

A very special thanks to Amitabh Sinha, Franklin J. Alier López de Haro, Hans-Jörg Bauer, Olle Nilsson, Stephan Rusch and Sudharsanan Selvaraj for their proofreading work.

¹ I'll pay you the original price in euros via PayPal. You have to provide me the corresponding corrections. This offer only applies to the latest eBook published edition

Contents

Chapter 1: Architecture & philosophy	1
1.1 The OpenXava concepts	2
Lightweight Model-Driven Development	2
Business Component	3
1.2 Application architecture	5
Application developer viewpoint	6
User viewpoint	7
Project layout	8
1.3 Flexibility	8
Editors	9
Custom view	9
1.4 Summary	10
Chapter 2: Java Persistence API	11
2.1 JPA Annotations	12
Entity	12
Properties	13
References	15
Embeddable classes	16
Collections	17
2.2 JPA API	19
2.3 Summary	21
Chapter 3: Annotations	23
3.1 Validation	24
Declarative validation	24
Built-in validations	24
Custom validation	26
Learn more about validations	27
3.2 User interface	28
The default user interface	28
The @View annotation	29
Refining member presentation	30
Learn more about the user interface	33
3.3 Other annotations	33
3.4 Summary	34

Chapter 4: Getting started with Eclipse and PostgreSQL	35
4.1 Our goal: A small Invoicing application	36
4.2 Installing PostgreSQL	36
4.3 Creating the project in Eclipse	38
Installing OpenXava	39
Create the project	39
Configuring Tomcat inside Eclipse	41
Creating your first entity	43
4.4 Preparing the database	45
Configuring persistence.xml	45
Update schema	46
4.5 Running the application	48
4.6 Modifying the application	49
4.7 Accessing a database from Eclipse	50
4.8 Summary	52
Chapter 5: Modeling with Java	53
5.1 Basic domain model	54
Reference (ManyToOne) as descriptions list (combo)	54
Stereotypes	56
Embeddable	57
Composite key	59
Calculating default values	61
Regular reference (ManyToOne)	63
Collection of dependent entities (ManyToOne with cascade)	64
5.2 Refining the user interface	66
Default user interface	67
Using @View for defining layout	67
Using @ReferenceView to refine the user interface for reference	68
Refining a collection item entry	69
Refined user interface	71
5.3 Agile development	71
5.4 Summary	74
Chapter 6: Automated testing	75
6.1 JUnit	76
6.2 ModuleTestBase for testing the modules	76
The code for the test	76
Executing the tests from Eclipse	78

6.3 Adding the JDBC driver to the Java Build Path	80
6.4 Creating test data using JPA	81
Using setUp() and tearDown() methods	81
Creating data with JPA	82
Removing data with JPA	84
Filtering data from list mode in a test	84
Using entity instances inside a test	85
6.5 Using existing data for testing	86
6.6 Testing collections	87
Breaking down tests in several methods	87
Asserting default values	89
Data entry	90
Verifying the data	91
6.7 Suite	92
6.8 Summary	93
Chapter 7: Inheritance	95
7.1 Inheriting from a mapped superclass	96
7.2 Entity inheritance	98
New Order entity	98
CommercialDocument as an abstract entity	99
Invoice refactored to use inheritance	100
Creating Order using inheritance	101
Naming convention and inheritance	102
Associating Order with Invoice	102
7.3 View inheritance	103
The extendsView attribute	103
View for Invoice using inheritance	104
View for Order using inheritance	106
Using @ReferenceView and @CollectionView to refine views	107
7.4 Inheritance in JUnit tests	109
Creating an abstract module test	109
Using the abstract module test to create concrete module tests	111
Adding new tests to the extended module test	111
7.5 Summary	113
Chapter 8: Basic business logic	115
8.1 Calculated properties	116
Simple calculated property	116
Using @DefaultValueCalculator	117

Calculated properties depending on a collection	120
Default value from a properties file	122
8.2 JPA callback methods	124
Multiuser safe default value calculation	124
Synchronizing persistent and calculated properties	125
8.3 Database logic (@Formula)	128
8.4 JUnit tests	130
Modifying existing tests	130
Testing default values and calculated properties	131
Testing calculated and persistent synchronized properties / @Formula	133
8.5 Summary	134
Chapter 9: Advanced validation	135
9.1 Validation alternatives	136
Adding delivered property to Order	136
Validating with @EntityValidator	137
Validating with a JPA callback method	139
Validating in the setter	139
Validating with Hibernate Validator	140
Validating on removal with @RemoveValidator	140
Validating on removal with a JPA callback method	142
What's the best way of validating?	142
9.2 Creating your own Hibernate Validator annotation	143
Using a Hibernate Validator from your entity	143
Defining your own ISBN annotation	144
Using Apache Commons Validator to implement the validation logic	144
Call to a REST web service to validate the ISBN	146
Adding attributes to your annotation	148
9.3 JUnit tests	149
Testing validation for adding to a collection	149
Testing validation assigning a reference and validation on removal	150
Testing the custom Hibernate Validator	151
9.4 Summary	152
Chapter 10: Refining the standard behavior	155
10.1 Custom actions	156
Typical controller	156
Refining the controller for a module	158
Writing your own action	159
10.2 Generic actions	162

MapFacade for generic code	162
Changing the default controller for all modules	163
Come back to the model a little	165
Metadata for more generic code	166
Chaining actions	168
Refining the default search action	169
10.3 List mode	173
Filtering tabular data	173
List actions	174
10.4 Reusing actions code	176
Using properties to create reusable actions	176
Custom modules	178
Several tabular data definitions by entity	179
Reusable obsession	179
10.5 JUnit tests	180
Testing the customized delete behavior	180
Testing several modules in the same test method	182
10.6 Summary	184
Chapter 11: Behavior & business logic	187
11.1 Business logic in detail mode	188
Creating an action for custom logic	189
Writing the real business logic in the entity	190
Write less code using Apache Commons BeanUtils	191
Copying a collection from entity to entity	192
Application exceptions	193
Validation from action	194
On change event to hide/show an action programmatically	195
11.2 Business logic from list mode	198
List action with custom logic	199
Business logic in the model over several entities	201
11.3 Changing module	203
Using IChangeModuleAction	203
Detail only module	204
Returning to the calling module	205
Global session object and on-init action	206
11.4 JUnit tests	209
Testing the detail mode action	209
Finding an entity for testing using list mode and JPA	211
Testing hiding of the action	212

Testing the list mode action	213
Asserting test data	215
Testing exceptional cases	215
11.5 Summary	216
Chapter 12: References & collections	217
12.1 Refining reference behavior	218
Validation is good, but not enough	218
Modifying default tabular data helps	219
Refining action for searching reference with list	220
Searching the reference typing in fields	222
Refining action for searching reference typing key	223
12.2 Refining collection behavior	226
Modifying default tabular data helps	226
Refining the list for adding elements to a collection	227
Refining the action to add elements to a collection	229
12.3 JUnit tests	232
Adapting OrderTest	232
Testing the @SearchAction	233
Testing the @OnChangeSearch	234
Adapting InvoiceTest	235
Testing the @NewAction	236
Testing the action to add elements to the collection	239
12.4 Summary	241
Chapter 13: Security & navigation with Liferay	243
13.1 Introduction to Java portals	244
13.2 Installing Liferay	245
13.3 Deploying our application on Liferay	246
13.4 Navigation	248
Adding the pages for the application and its modules	248
Filling an empty page with a Portlet	250
Adding left navigation menu	250
Copying page	251
Two modules in the same page	252
Using CMS	253
13.5 User management	256
Roles	256
Users	257
13.6 Access levels	258

Limiting access to the entire application	258
Limiting access to individual modules	260
Limiting functionality	261
Limiting list data visibility	261
Limiting detail data visibility	263
13.7 JUnit tests	265
13.8 Summary	267
Appendix A: Source code	269
Appendix B: Learn more	317